Design and Implementation of a Workflow Rendering Engine

Jason A. Pamplin   Ying Zhu

Georgia State University
Atlanta, Georgia

Abstract

Business visualization is an effective tool for business decision making and problem solving. Despite its obvious benefits, 3D visualization is far less common in practice than it should be due to the lack of software support. To address this problem, we have developed a workflow rendering engine that facilitates the 3D visualization of business process information for organizational process analysis and efficiency studies. The workflow information is represented in XML format to be compatible with the new Web graphics standard X3D. The rendering engine is implemented using C++ and OpenGL in order to take full advantage of modern 3D graphics hardware. The object-oriented design makes this engine highly extensible.

Keywords: *business visualization, workflow, rendering engine, XML, OpenGL,*

1. Introduction[1]

In many organizational situations, it is desirable to be able to look at the business processes and determine where the main inefficiencies are. Such processes are often referred to as “workflow”. Currently, most workflow diagrams are rendered and manipulated solely in two dimensions. This is because the software packages that draw such diagrams lack the support for 3D visualization.

3D graphical representation of abstract business data has a number of benefits over 2D representations, or rows and columns of numbers. The added dimension allows user to combine more information into a single scene. Complex information can therefore be conveyed in an easy-to-understand manner. 3D visualization also allows for more flexible and sophisticated computer-human interactions.

In this paper, we present the design and implementation of a workflow rendering engine that can be used for 3D visualization and manipulation of workflow information in organizational process analysis and efficiency studies. Workflow is represented in XML format so that the information can be visualized over the Web. The rendering engine is implemented using C++ and OpenGL in order to take full advantage of modern 3D graphics hardware.

The paper is organized as follows. In section 2 we discuss the background of business information visualization and related work. Section 3 and 4 focus on the workflow representation and rendering engine design. Section 5 presents our experiment result. Section 6 concludes the paper.

2. Background

Business information visualization is a relatively new field but has gained adoption among business users because it supports a variety of business tasks, including decision-making, problem solving, knowledge management, and business performance management [2][7]. It has been shown that visualization can aid business decision makers to deal with information overflow and therefore enhance their problem solving capabilities [5]. The positive impact of business visualization has been felt in many industries, including finance, marketing, retail, manufacturing, consumer product goods [1][3][4][5][6][8].

Improving business process is a common theme in many organizations. Decision makers often want to review the process and determine where the main inefficiencies are. Business process is often visualized using workflow diagrams, where business actions are depicted as 2D artifacts linked by arrows that indicate the direction of information flow. Figure 1 shows a sample 2D business workflow diagram.

[1] The contact author is

The amount of information that can be presented in two-dimensional space is limited. The user manipulation of the 2D workflow diagram is also limited. A key feature of effective business visualizations is to allow users to easily manipulate different views [2]. Therefore, it is desirable to expand the workflow visualization to 3D space.

Schonhage et al. [4] have presented a 3D business process visualization program developed using Java3D. The business process is represented using VRML format, which is already superceded by a newer XML based Web graphics standard X3D. The program is developed for the process visualization of a specific organization and therefore the workflow and rendering are closely coupled. This makes it unsuitable for generic workflow visualization.

We have designed and implemented a workflow rendering engine for business process visualization. The business process is stored in a XML format file, making it compatible with the new X3D standard and suitable for Web application. At runtime, the XML file is processed and translated into a scene data structure appropriate for visualization. The rendering part and workflow representation is therefore decoupled for flexibility and information hiding.

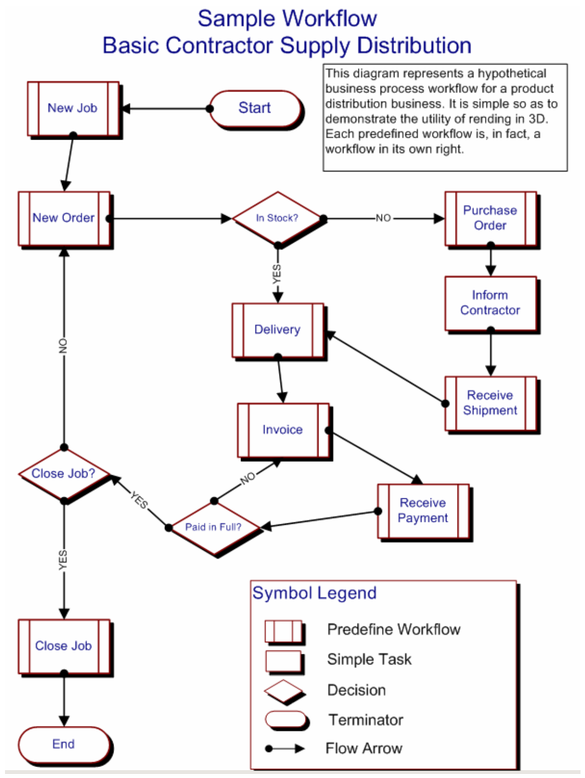
Sample Workflow
Basic Contractor Supply Distribution
This diagram represents a hypothetical business process workflow for a product distribution business. It is simple so as to demonstrate the utility of rending in 3D. Each predefined workflow is, in fact, a workflow in its own right.
Start
New Job
New Order
In Stock?
NO
Purchase Order
YES
Inform Contractor
Delivery
NO
Receive Shipment
Invoice
Close Job?
NO
Receive Payment
YES
Paid in Full?
YES
Close Job
End
Symbol Legend
Predefine Workflow
Simple Task
Decision
Terminator
Flow Arrow

Figure 1 2D representation of a sample workflow

3. Workflow representation

The XML specification of workflow consists of a task section and a flow section. Flows are the objects which connect two tasks together. Appendix A contains the XML specification for the workflow diagram in Figure 1.

Each task and flow is assigned a unique ID. A task has a type which represents one of the four acceptable types for a task in the diagram. Each task also has an *x* and *y* coordinate which corresponds to its physical location in the *x* and *y* plane. Note that the specification comes from a 2D diagram so the XML specification does not keep any *z* axis information. This is important as we are reserving the use of the *z* axis to display statistical information.

Each task also has a name which should be unique across all tasks and should be descriptive of what the task is. The name should be limited to no more than 20 characters. Spaces in the name are allowed and encouraged. Finally, we can add many statistical information fields. In the example shown in section 5, we only use the *average_time* field. The *flow* tag simply has an ID and additional child tags which indicate from which task and to which task the flow is going. The tags are labeled “from” and “to” in order to provide directional information.

4. Workflow rendering engine design

The rendering engine uses the following libraries:

- Xercesc – This is the Apache Xerces C++ XML implementation. This is used for parsing the XML file.
- Xalanc – This is also an Apache module for C++ for XML and XSL translations.
- GLUT – the Graphics Library Utility Toolkit for windowing and menu system creation.
- OpenGL – the 3D Graphics Library.

The engine is designed based on object-oriented principles. There are a series of *Objects* that are all derived from the *Base* class. There is also a *Scene* object inherited from the *Base* class as well. Basically, a *Scene* is itself a repositionable object that contains a collection of *Lights* and *Objects*. The *Scene* simply manages these objects allowing limited access to them during the course of program execution.

This model is extended for use in the workflow rendering. We create a *Task* base class that is inherited from the *Object*. Then each type of task is extended from it. The *Flow* class is also extended from the *Object* with additional data points and methods for calculating the orientation and position based on two end-points. The *Flow* and *Task* objects have additional code for writing their own labels during rendering.

Likewise, the *Workflow* class was extended from the *Scene* since tasks and flows are collections of *Objects*.

The class diagram of the rendering engine is shown in Figure 2.

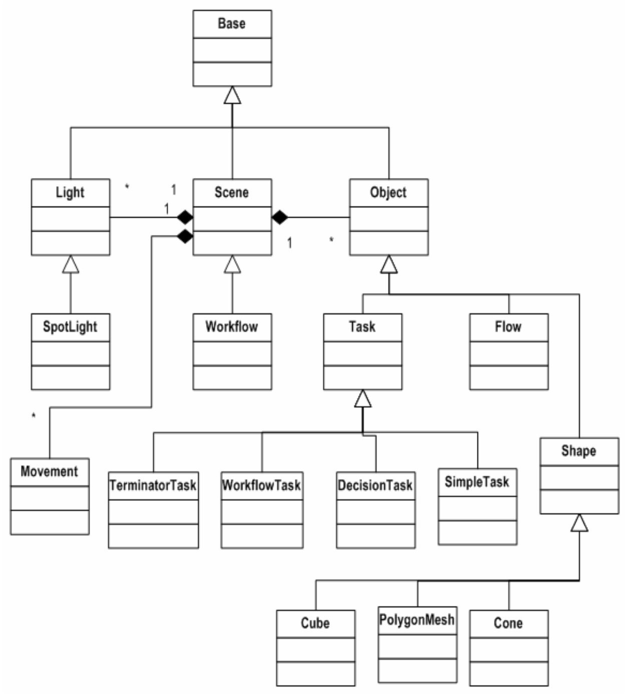

Figure 2 Class diagram of the rendering engine

5. Result

The rendering engine is developed using C++ and OpenGL. Object-oriented design makes the rendering engine highly extensible. For example, we can easily add new tasks, flow, and graphics elements. Compared with Java3D, OpenGL can take full advantage of the latest graphics hardware and therefore making the rendering much more efficient.

The program can be run from the command line with one command line argument which should be the path to the XML document that contains the workflow information.

The following snapshots (Figure 3, 4 and 5) are taken from 3D visualization of the same workflow depicted in Figure 1.

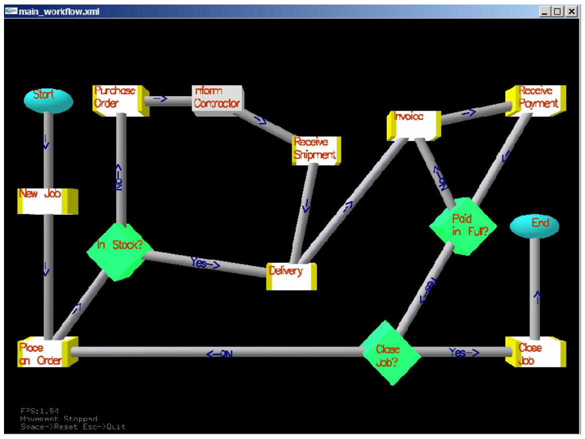

Figure 3 3D visualization of workflow information

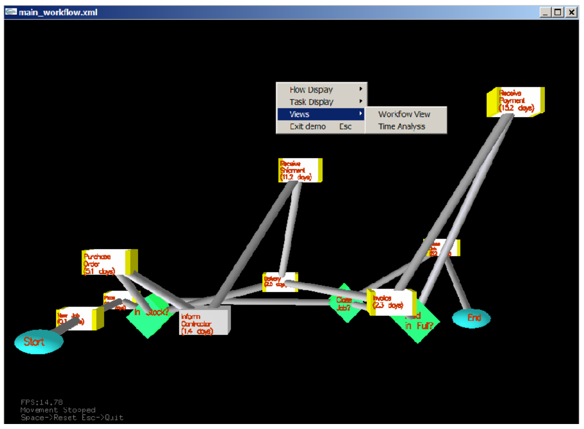

Figure 4 Time analysis view

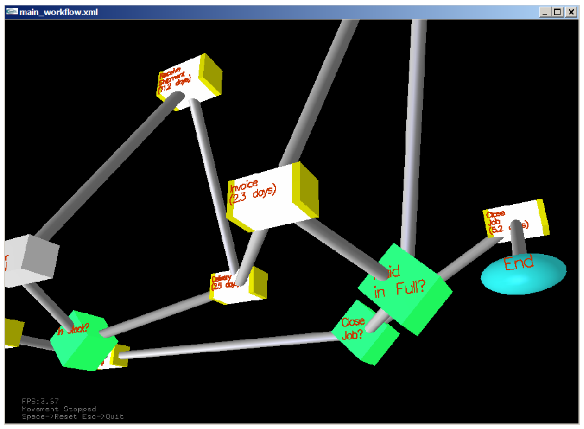

Figure 5 Workflow can be viewed from any angle

In this example, user can easily switch between workflow view and time analysis view. The rendering engine smoothly transforms the 3D representation from one view to the other. User can freely walk through the 3D workflow structure, viewing the workflow information from any angle, and manipulating it in 3D space. This makes it extremely easy to identify the slow points in the workflow, particularly in the Time Analysis View.

6. Conclusion and future work

Business visualization is an effective tool for business decision making and problem solving. Despite its obvious benefits, 3D visualization is far less common in practice than it should be due to the lack of software support. To address this problem, we have developed a rendering engine that facilitates the 3D visualization of business process information for time analysis and efficiency improvement.

In our design, the workflow information is represented in XML format to be compatible with the new Web graphics standard X3D. The rendering engine is implemented using C++ and OpenGL in order to take full advantage of modern 3D graphics hardware. The object-oriented design makes this engine highly extensible.

In our 3D visualization, users can smoothly switch among different views. The 3D workflow structure is easy to read in either orientation, making it extremely easy to identify the slow points in the workflow.

What we have presented in this paper is only the first step towards a Web based, generic 3D workflow visualization environment. In the future, we are planning to improve our workflow rendering engine in the following aspects.

- Drill down a workflow to see its details, allowing the user to navigate through workflow details.
- Zoom in on a specific task and see more detail about it.
- Add additional classifications such as permissions, etc. The 3-D space could be used for grouping based on attributes as well.
- Allows for more flexible user manipulation of the diagram in 3D space.
- Develop an intelligent algorithm for placing objects in the most visible spot.

7. References

[1] R. B. Dull and D. P. Tegarden, "Visualization of Complex Multi- Dimensional Accounting Information." in Proceedings of the Fourth Americas Conference on Information Systems, Ellen D. Hoadley and Izak Benbasat (eds.), AIS, pp. 6-8, 1998.

[2] Blender3D, http://www.blender3d.com/.

[3] B. Schonhage, A. van der Scheer, E. Treur, and A. Eliens, Visualization and Simulation of Business Information at Gak NL. Workshop on New Paradigms in Information Visualization and Manipulation 1999.

[4] B. Schonhage, A. van Ballegooij, and A. Elliens, 3D Gadgets for Business Process Visualization – a Case Study, Proceedings of the 5th symposium on Virtual Reality Modeling Language, Monterey, California, pp. 131 – 138, 2000.

[5] D. P. Tegarden, Business Information Visualization, Communications of the Association for Information Systems, Vol. 1, Paper 4, 1999.

[6] W. Wright, Business Visualization Applications, IEEE Computer Graphics and Applications, Vol. 17, No. 4, pp. 66-70, 1997.

[7] P. Zhang, Business Information Visualization: Guidance for Research and Practice, Encyclopedia of Microcomputers, Volume 27, 2001.

[8] P. Zhang and D. Zhu, Information Visualization in Project Management and Scheduling, Proceedings of The 4th Conference of the International Society for Decision Support Systems (ISDSS'97), Ecole des HEC, University of Lausanne, Switzerland, July 21-22, 1997.

8. Appendix A – XML representation of the workflow depicted in Figure 1.

```
<?xml version="1.0" encoding="Windows-1252" ?>
<workflow>
 <tasks>
  <task id="1" type="terminator" x="-10" y="5">
   <name>Start</name>
  </task>
  <task id="2" type="workflow" x="-10" y="1">
   <name>New Job</name>
   <average_time>.3</average_time>
  </task>
  <task id="3" type="workflow" x="-10" y="-5">
   <name>Place an Order</name>
   <average_time>.2</average_time>
  </task>
  <task id="4" type="decision" x="-7" y="-1">
```

```
    <name>In Stock?</name>
    <yes>11</yes>
    <no>10</no>
  </task>
  <task id="5" type="workflow" x="-7" y="5">
    <name>Purchase Order</name>
    <average_time>5.1</average_time>
  </task>
  <task id="6" type="workflow" x="0" y="-2">
    <name>Delivery</name>
    <average_time>2.5</average_time>
  </task>
  <task id="7" type="simple" x="-3" y="5">
    <name>Inform Contractor</name>
    <average_time>1.4</average_time>
  </task>
  <task id="8" type="workflow" x="1" y="3">
    <name>Receive Shipment</name>
    <average_time>11.2</average_time>
  </task>
  <task id="9" type="workflow" x="5" y="4">
    <name>Invoice</name>
    <average_time>2.3</average_time>
  </task>
  <task id="10" type="workflow" x="10" y="5">
    <name>Receive Payment</name>
    <average_time>15.2</average_time>
  </task>
  <task id="11" type="decision" x="7" y="0">
    <name>Paid in Full?</name>
  </task>
  <task id="12" type="decision" x="4" y="-5">
    <name>Close Job?</name>
  </task>
  <task id="13" type="workflow" x="10" y="-5">
    <name>Close Job</name>
    <average_time>5.2</average_time>
  </task>
  <task id="14" type="terminator" x="10" y="0">
    <name>End</name>
  </task>
</tasks>
<flows>
  <flow id="20">
    <from>1</from>
    <to>2</to>
  </flow>
  <flow id="21">
    <from>2</from>
    <to>3</to>
  </flow>
  <flow id="22">
    <from>3</from>
    <to>4</to>
  </flow>
  <flow id="23" label="No">
    <from>4</from>
    <to>5</to>
  </flow>
  <flow id="24" label="Yes">
    <from>4</from>
    <to>6</to>
  </flow>
  <flow id="25">
    <from>5</from>
    <to>7</to>
  </flow>
  <flow id="26">
    <from>7</from>
    <to>8</to>
  </flow>
  <flow id="27">
    <from>8</from>
    <to>6</to>
  </flow>
  <flow id="28">
    <from>6</from>
    <to>9</to>
  </flow>
  <flow id="29">
    <from>9</from>
    <to>10</to>
  </flow>
  <flow id="30">
    <from>10</from>
    <to>11</to>
  </flow>
  <flow id="31" label="No">
    <from>11</from>
    <to>9</to>
  </flow>
  <flow id="32" label="Yes">
    <from>11</from>
    <to>12</to>
  </flow>
  <flow id="33" label="No">
    <from>12</from>
    <to>3</to>
  </flow>
  <flow id="34" label="Yes">
    <from>12</from>
    <to>13</to>
  </flow>
  <flow id="35">
    <from>13</from>
    <to>14</to>
  </flow>
 </flows>
</workflow>
```